AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions and listings of claims in the application:

Listing of Claims:

1. (**Currently Amended**) A method of optimizing the performance of an interpreter-based runtime system, ~~said runtime~~the runtime system including a virtual machine, the virtual machine adapted to run an application in the context of the runtime environment, the method comprising:

augmenting the bytecode set of the virtual machine with application-specific ~~codes~~opcodes ~~by reference to said application~~, thereby constituting an application domain-specific virtual machine;

optimizing the virtual machine based on semantics of the application to be run on the virtual machine;

performing a quantitative trade-off between time and space and encoding semantically enriched code based upon the trade-off;

analyzing frequently executed codes and encoding semantically enriched codes of the instruction set of the virtual machine to efficiently decode the frequently executed codes; and

statically embedding the semantically enriched code to optimize execution of the runtime system on the application domain-specific virtual machine.

2. (Previously Presented) The method as claimed in claim 1 wherein the virtual machine is a Java Virtual Machine.

3. (Previously Presented) The method as claimed in claim 1 wherein a new application domain-specific virtual machine is generated for different categories of applications.

4. (Previously Presented) The method as claimed in claim 1 wherein the dynamic and/or static behavior of the application is used to create new opcode for the application domain-

specific virtual machine.

5. **(Canceled)**.

6. (Previously Presented) The method as claimed in claim 1 wherein the virtual machine is optimized based on a late-binding or dynamic loading model and runtime constant manifestation.

7. **(Canceled)**.

8. **(Currently Amended)** The method as claimed in claim 1 wherein semantically enriched code is dynamically embedded to enable it to run fast on the application domain-specific virtual machine, said virtual machine newly generated in accordance with claim 1.

9. **(Canceled)**.

10. **(Currently Amended)** The method as claimed in ~~claim 7~~claim 1 wherein the semantically enriched code is determined based on the dynamic and/or static behavior of the application.

11. **(Currently Amended)** ~~The method~~A method of generating an embedded virtual machine for a specific domain of an application, comprising the step of:

_____embedding semantically enriched code in an interpreter loop of the virtual machine to efficiently decode frequently executed code.

12. (Previously Presented) The method as claimed in claim 11 wherein the semantically enriched code embedding step is performed dynamically on newly loaded portions of the application in dynamic languages.

13. (Previously Presented) The method as claimed in claim 12 wherein the interpreter is dynamically enhanced.

14. (Previously Presented) The method as claimed in claim 11 wherein secondary codes are used to accommodate the interpretation of new semantically enriched codes.

15. **(Canceled)**.

16. (Previously Presented) The method as claimed in claim 14 wherein if a particular code is used frequently, it is made into a single byte code and the rest of the semantically enriched codes are accommodated by secondary codes.

17. **(Currently Amended)** A method of optimizing the performance of an application running on an interpreter-based runtime system, the method comprising:

augmenting ~~the bytecode~~a bytecode set of the interpreter with new semantically enriched application-specific opcodes by reference to said application, thereby constituting an application domain-specific virtual machine;

wherein the encoding of new semantically enriched opcodes efficiently decodes frequently executed codes.

18. (Canceled).

19. (Canceled).

20. (Previously Presented) A computer program, recorded on a computer-readable medium, to perform the method as claimed in claim 17.